SELÇUK ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

3302659-3312659
GÖRÜNTÜ İŞLEME (YM)

PROJE 2

DERSİN HOCASI

Dr. Kürşad UÇAR

Enes ÇAKIR
203312011

Proje Tarihi	Rapor Teslim Tarihi	Proje Notu
14/04/2025	24/04/2025	

1. PROJENİN AMACI

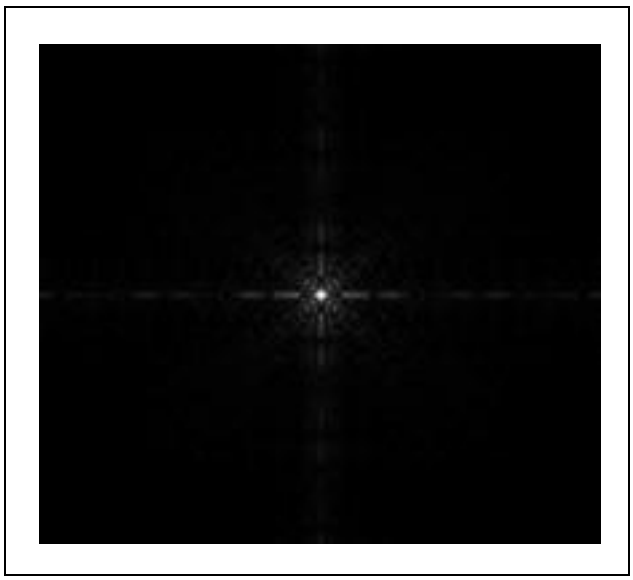
Projenin amacı, temel görüntü işleme tekniklerinden biri olan noktasal işleme yöntemlerini uygulayarak gri tonlu ve renkli görüntüler üzerinde iyileştirme işlemleri gerçekleştirmektir. Bu kapsamda; logaritmik **dönüşüm (log transform)**, **power-law transform**, **gri seviye dilimleme (gray-level slicing)**, **bit düzlemi dilimleme (bitplane slicing)**, **histogram eşitleme** ve **kontrast germe (contrast stretching)** gibi teknikler kullanılarak görüntülerin kontrastı artırılmakta, önemli detayların öne çıkarılması sağlanmakta ve görüntü kalitesi iyileştirilmektedir.

2. TEST SONUÇLARI

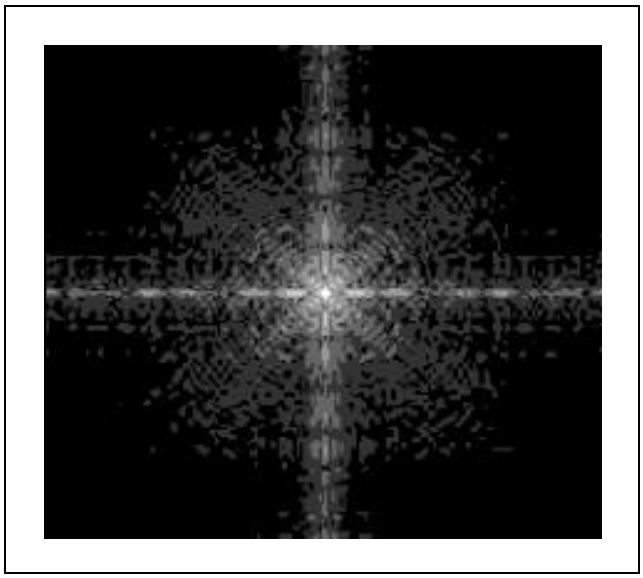
2.1 Log Transform ve Power-Law Sonuçları

Log dönüşümü, düşük yoğunluklu piksellerin kontrastını artırmakta etkiliyken; **power-law dönüşümü**, gama (γ) parametresi sayesinde daha esnek bir parlaklık ve kontrast kontrolü sağlamaktadır. Gama değeri 1'den küçük olduğunda görüntü aydınlanır, büyük olduğunda ise karartılır. Her iki yöntemin temel amacı kontrastı artırarak görüntüdeki detayları belirginleştirmektir. Benzer yönleri, her ikisinin de doğrusal olmayan dönüşümler olması ve giriş piksellerini logaritmik ya da üs fonksiyonlarıyla yeniden ölçeklendirmeleridir. Ancak log dönüşüm sabit bir matematiksel yapı sunarken**, power-law dönüşümü** gama parametresiyle daha fazla kontrol imkânı sağlar. Bu nedenle, **log dönüşümün** yerine **power-law transform** çoğu durumda tercih edilebilir, çünkü daha geniş bir uygulama esnekliği sunar.

Log Transform Görüntüleri



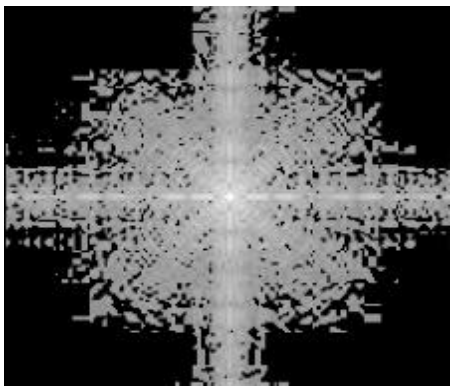
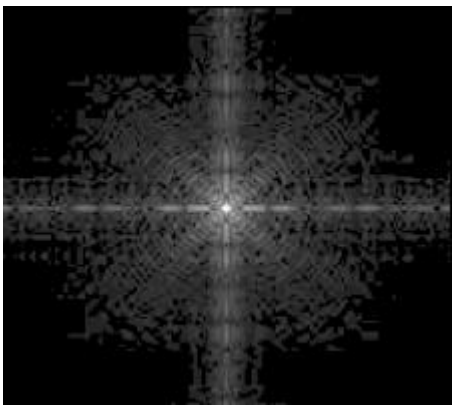
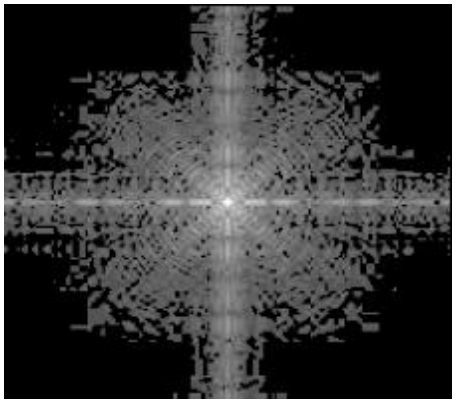
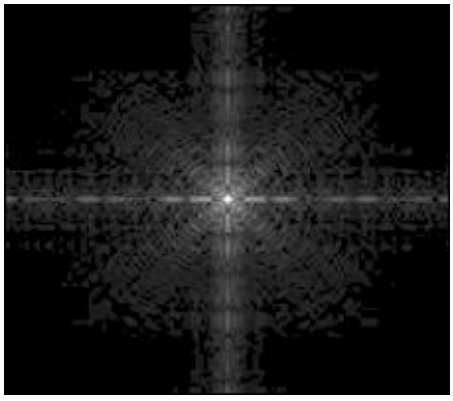
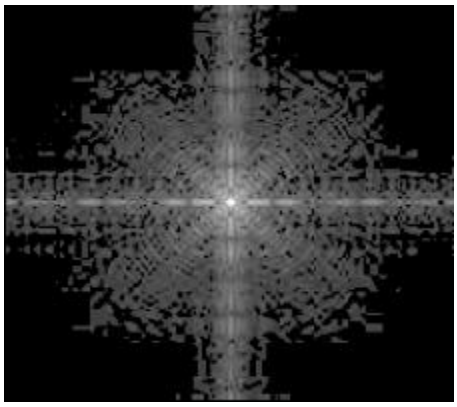
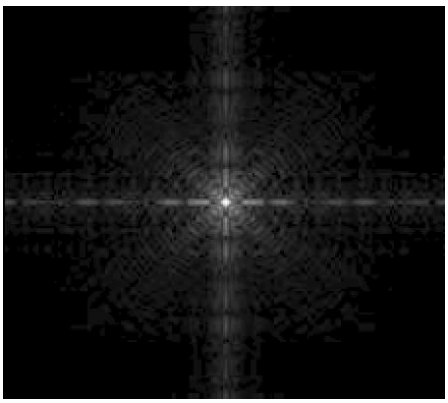
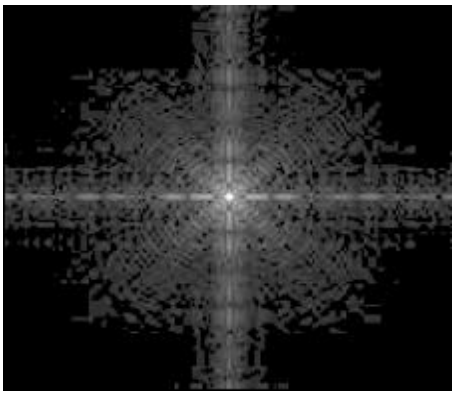
Orijinal Görüntü



Log Transform sonucu oluşan Görüntü

Power-Law Görüntüleri ve Gama seçimi

Gama değerleri 0.1 ve 0.6 değerlerden oluşmaktadır.

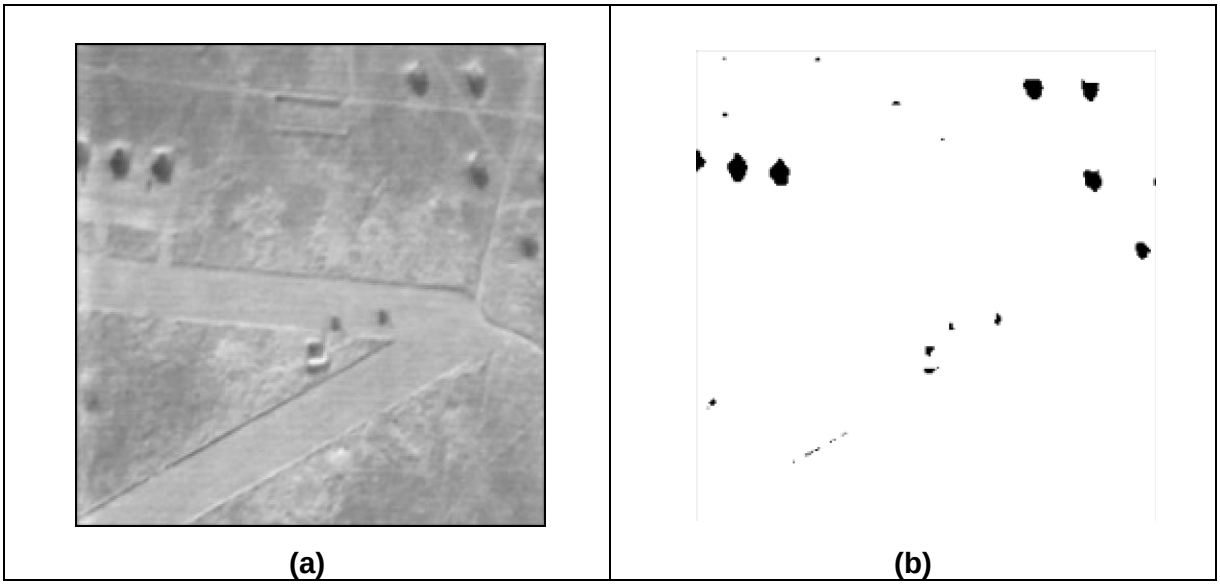









2.2 Gray-level slicing

Görüntüde belirli bir piksel aralığındaki detayları vurgulamak amacıyla gray-level slicing (gri seviye dilimleme) algoritması uygulanmıştır. Bu yöntem, özellikle belirli parlaklık seviyesine sahip nesneleri öne çıkarmak için kullanılır. Projede, sığınak.jpg görüntüsü üzerinde uygun bir eşik değeri belirlenerek sadece belirli piksel aralığında kalan bölgeler vurgulanmıştır. Gray-level slicing yöntemi sayesinde belirlenen aralıktaki pikseller parlak hale getirilirken, diğer pikseller karartılmıştır. Bu yöntem hem manuel olarak belirlenen eşik değerleriyle hem de otomatik eşikleme yöntemleri ile uygulanabilir. Görselde manuel olarak eşik seviye uygulanmıştır ve sığınakları belirginleştirmek amacıyla eşik seviyesi 81 olarak seçilmiştir.

Peki eşik seviyesi nedir? Eşik seviyesi, bir görüntüdeki piksellerin parlaklık (yoğunluk) değerlerine göre sınıflandırılmasını sağlayan sınır değeridir. Gri tonlu görüntülerde her piksel 0 ile 255 arasında bir parlaklık değeri taşır. Bu eşik değerine göre pikseller iki gruba ayrılır: Eğer bir pikselin parlaklık değeri eşik seviyesinden büyükse genellikle beyaz (255), küçükse siyah (0) olarak atanır. Böylece görüntü ikili hale getirilerek belirli nesneler ya da bölgeler vurgulanabilir. Bu işlem, nesne ve arka plan ayrımı gibi birçok görüntü işleme uygulamasında yaygın olarak kullanılmaktadır.

Otomatik eşikleme nasıl uygulanır? Otomatik eşikleme için histogramdan yararlanabiliriz. Histogram, bir görüntüdeki tüm piksel değerlerinin frekans dağılımını gösterir. Bu dağılım analiz edilerek, görüntüdeki nesne ve arka plan arasındaki kontrast farklarına göre ideal bir eşik değeri belirlenebilir. Örneğin, histogram iki belirgin tepe içeriyorsa, bu tepeler arasında kalan en düşük noktadaki yoğunluk değeri eşik seviyesi olarak seçilebilir. Bu yönteme Otsu yöntemi denir ve yaygın olarak kullanılır. Otsu algoritması, sınıf içi varyansı minimize ederek görüntüyü iki farklı bölgeye en uygun şekilde ayıracak eşik değerini otomatik olarak hesaplar. Böylece, kullanıcı müdahalesi olmadan görüntüdeki nesneler daha doğru şekilde ayrıştırılabilir.

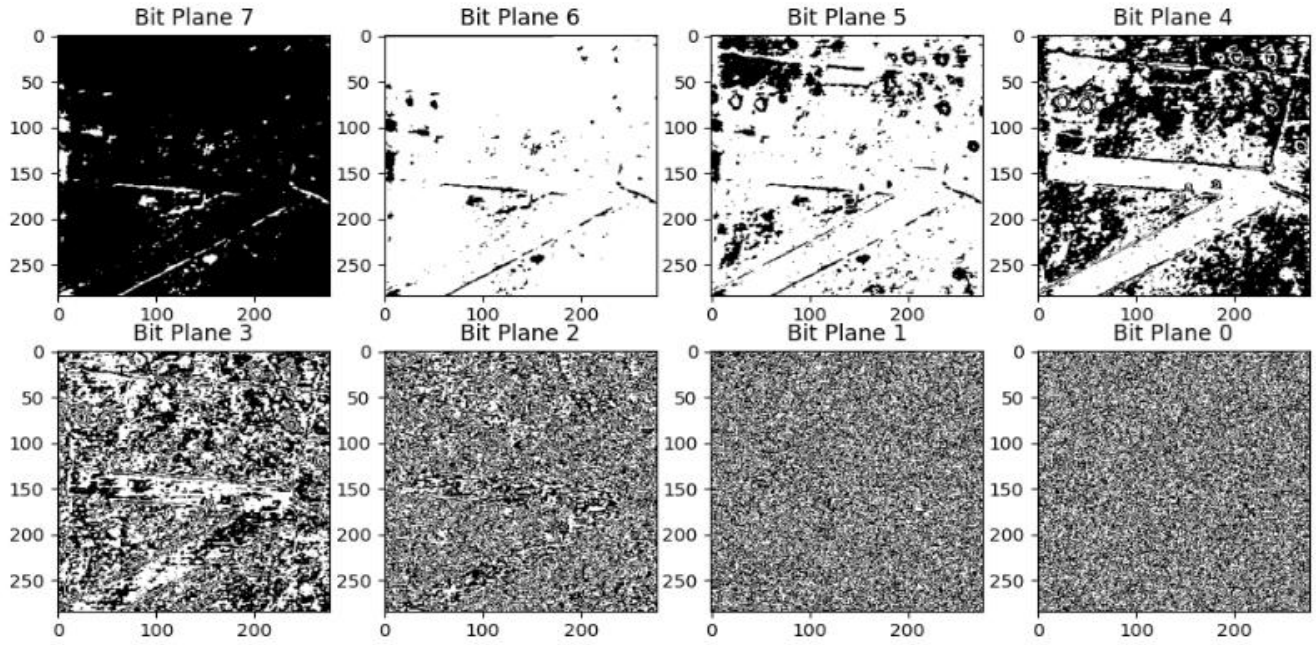


(a) **(b)**

a)Orijinal Görüntü, b)Eşik Seviyesi 81 olarak uygulanmış görüntü

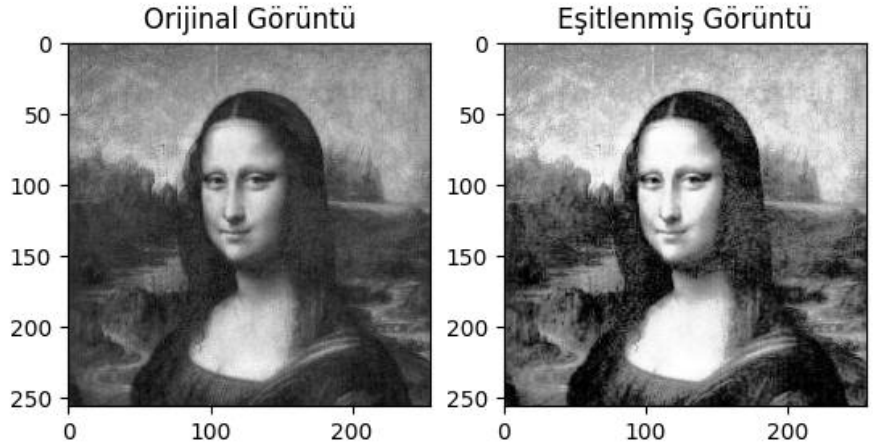
2.3 Bitplane slicing

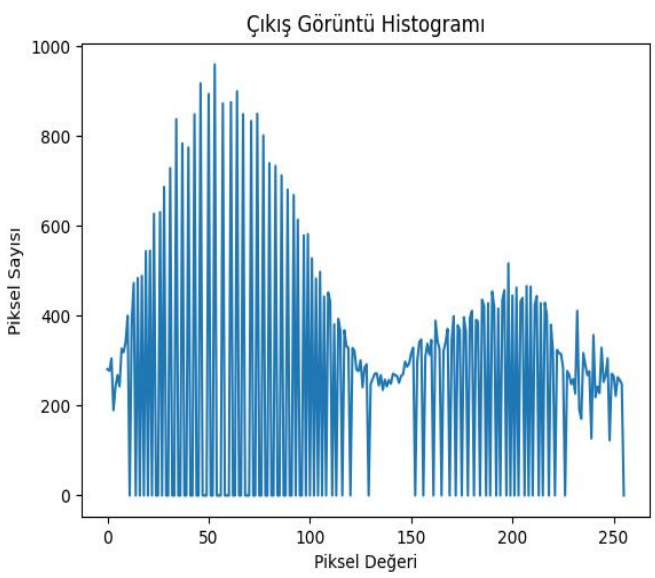
Projede, bitplane slicing algoritması kullanılarak *sığınak.jpg* görüntüsü üzerinde detaylı analiz yapılmıştır. Bitplane slicing yöntemi, bir gri seviye görüntüyü oluşturan piksellerin her bir bit düzeyini ayrı ayrı ayırarak, görüntüyü sekiz farklı düzleme ayırır. Her düzlem, piksel değerlerinin belirli bir bitini temsil eder. Bu sayede görüntüdeki hangi detayların hangi bit düzeyinde yer aldığı gözlemlenebilir. Görselde, en düşük bit düzeyleri daha az bilgi içerirken, en yüksek (en soldaki, yani MSB - Most Significant Bit) bit düzeyi genellikle görüntünün en çok bilgisini taşır. Çünkü bu bit, pikselin değerine en fazla katkı sağlayan bittir. Ancak bu durum görüntü içeriğine göre değişiklik gösterebilir; bazen orta seviyedeki bitlerde önemli detaylar yer alabilir. Bu yöntem, özellikle gizli veri gömme (steganografi) ve görüntü sıkıştırma gibi uygulamalarda oldukça faydalıdır.

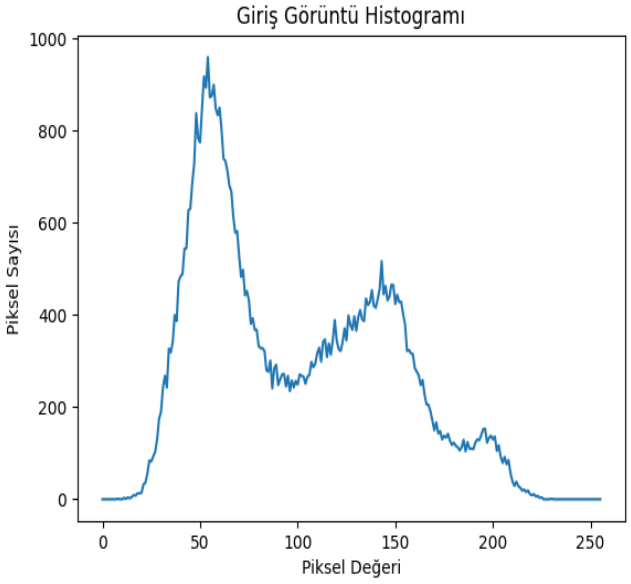


2.4 Histogram Eşitleme

Histogram eşitleme algoritması uygulanarak gri tonlu bir görüntünün kontrastı artırılmıştır. monalisa.jpg görüntüsü kullanılarak yapılan bu işlemde, önce orijinal görüntünün histogramı hesaplanmış, ardından her piksel için yeni değerler belirlenerek eşitlenmiş bir görüntü elde edilmiştir. Elde edilen sonuçlarda giriş ve çıkış görüntülerinin ortalama ve standart sapmaları hesaplanarak karşılaştırılmıştır. Son olarak hem orijinal hem de eşitlenmiş görüntü ile bu görüntülere ait histogramlar görsel olarak sunulmuştur. Bu yöntem sayesinde görüntüdeki detaylar daha belirgin hale getirilmiştir.







Giriş Görüntüsü Ortalama: 0.0027923583984375

Giriş Görüntüsü Standart Sapma: 108.90141557251819

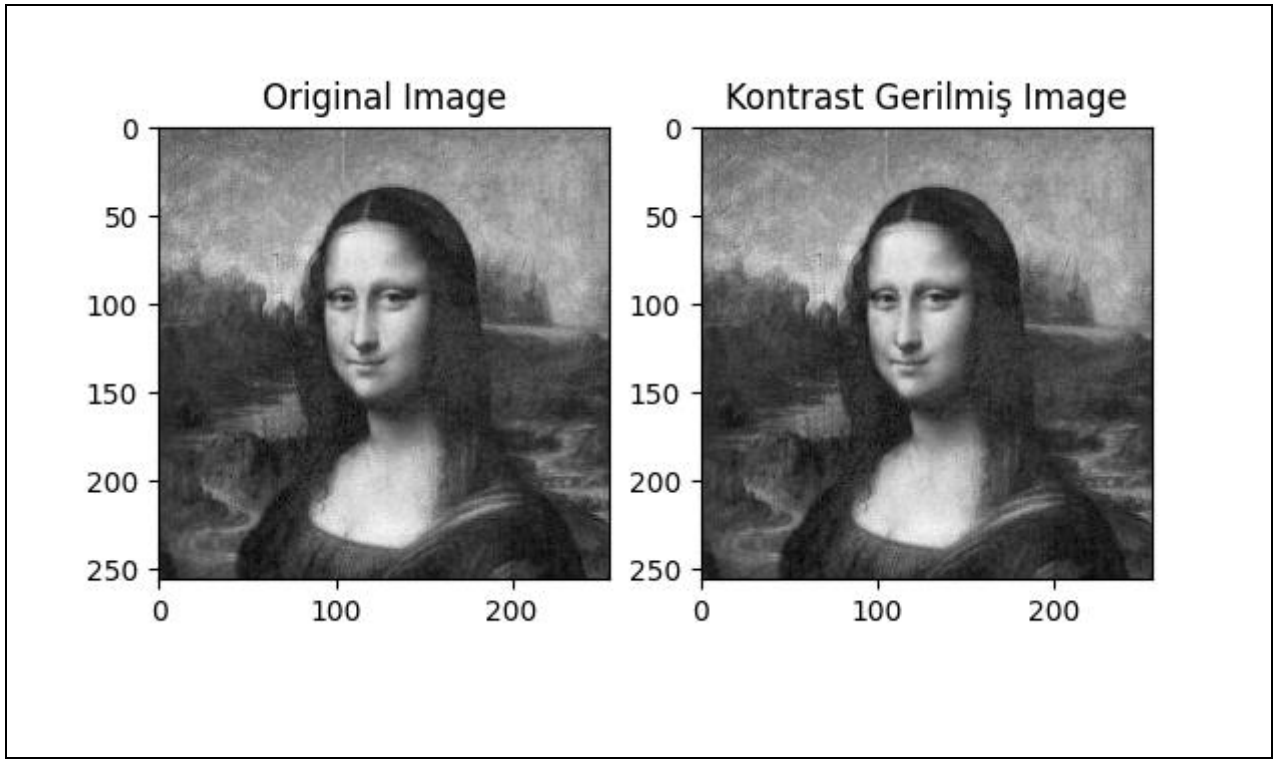
Çıkış Görüntüsü Ortalama: 0.00146484375

Çıkış Görüntüsü Standart Sapma: 146.1652860223766

2.5 Kontrast germe (contrast stretching)

Kontrast germe (contrast stretching), bir görüntüdeki en düşük ve en yüksek piksel değerlerini alıp bunları tam dinamik aralığa (örneğin 0–255) yayarak kontrastı artıran bir tekniktir. Bu işlem sayesinde, görüntüdeki koyu alanlar daha koyu, parlak alanlar daha parlak hale gelir ve detaylar daha belirgin olur. Görüntünün histogramı genişletilerek daha dengeli bir parlaklık dağılımı sağlanır. Bu yöntem, görüntünün doğal yapısını koruyarak kontrast iyileştirmesi yapar.

Kontrast germe ve histogram eşitleme yöntemleri karşılaştırıldığında; kontrast germe, daha basit ve doğal sonuçlar üretirken, histogram eşitleme görüntünün tüm ton aralığını eşit dağıtmaya çalışır ve bu sayede düşük kontrastlı alanlarda detayları daha fazla ortaya çıkarabilir. Ancak histogram eşitleme bazı durumlarda görüntünün doğal yapısını bozabilir ve aşırı parlaklık-karanlık bölgeler oluşabilir. Kontrast germe, görüntünün yapısına daha sadık kalır fakat her zaman yeterli kontrast artışı sağlamayabilir. Bu nedenle, bu iki yöntem birbirinin yerine her zaman kullanılamaz; görüntünün ihtiyaçlarına göre seçim yapılmalıdır.



3.1 Log Transform ve Power-Law kodları

```
#Log Transform
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Gri seviye bir örnek görüntü yükleme
inimage = cv2.imread('fourirespectrum.jpg', cv2.IMREAD_GRAYSCALE)

row, col = inimage.shape
log_transformed = np.zeros((row, col), dtype=np.uint8)
# Logaritmik dönüşüm

for x in range(row):
    for y in range(col):
        log_transformed[x, y] = 1 * (np.log(inimage[x, y] + 1))
#c = 255 / np.log(1 + np.max(inimage))
#log_transformed = 2 * (np.log(inimage + 1))

# Görüntüleri gösterme
plt.subplot(1, 2, 1), plt.imshow(inimage, cmap='gray'), plt.title('Original Image')
plt.subplot(1, 2, 2), plt.imshow(log_transformed, cmap='gray'), plt.title('Log
Transformed Image')
plt.show()

#PowerLaw
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Gri seviye bir örnek görüntü yükleme
inimage = cv2.imread('fourirespectrum.jpg', cv2.IMREAD_GRAYSCALE)

gamma = 0.6 # Güç yasası parametresi

# Power-Law Transformation işlemi
power_law_transformed = np.power(inimage, gamma)

# Görüntüleri gösterme
plt.subplot(1, 2, 1), plt.imshow(inimage, cmap='gray'), plt.title('Original Image')
plt.subplot(1, 2, 2), plt.imshow(power_law_transformed, cmap='gray'), plt.title('Kuvvet
Alınmış Image')
plt.show()
```

3.2 Gray level Slicing Kodu

```
#graylevelSlicing
import numpy as np
import matplotlib.pyplot as plt
import cv2

# Görüntüyü oku
inimage = cv2.imread("siginak.jpg", cv2.IMREAD_GRAYSCALE)

# Eşikleme fonksiyonu
def esikleme(goruntu):
    h, w = goruntu.shape
    esikleme_zeros = np.zeros((h, w), dtype=np.uint8)
    for i in range(h):
```

```
        for j in range(w):
            if goruntu[i, j] >= 81: #eşik değeri
                esikleme_zeros[i, j] = 255
            else:
                esikleme_zeros[i, j] = 0
    return esikleme_zeros

# fonksiyon çağırılıyor
cikis_goruntu = esikleme(inimage)

# Görüntüleri göster
plt.subplot(1, 2, 1)
plt.imshow(inimage, cmap='gray')
plt.title("Orijinal Görüntü")

plt.subplot(1, 2, 2)
plt.imshow(cikis_goruntu, cmap='gray')
plt.title("Eşiklenmiş Görüntü")
plt.show()
```

3.3 Bitplane Slicing Kodu

```
#BitplaneSlicing
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Gri seviye bir örnek görüntü yükleme
inimage = cv2.imread('siginak.jpg', cv2.IMREAD_GRAYSCALE)

# Görüntü boyutları
height, width = inimage.shape
print(height, width )
bit_planes = np.zeros((height, width,8))
# Bit Plane Slicing işlemi
for kaydirma in range(8):
    mask = 1 << kaydirma
    for i in range(height):
        for ii in range(width):
            bit_value = (inimage[i, ii] & mask) >> kaydirma
            bit_planes[i,ii,kaydirma] = bit_value


# Görüntüleri gösterme
plt.figure(figsize=(12, 6))
plt.subplot(2, 4, 1), plt.imshow(inimage, cmap='gray'), plt.title('Original Image')

for i in range(7, -1, -1):
    plt.subplot(2, 4, 8 - i), plt.imshow(bit_planes[:,:,i], cmap='gray'),
plt.title(f'Bit Plane {i}')

plt.show()
```

3.4 Histogram Eşitleme, Ortalama değer ve Standart sapmaları

```
#
import cv2
import matplotlib.pyplot as plt
import numpy as np

# Görüntüyü oku
inimage = cv2.imread('monalisa.jpg', cv2.IMREAD_GRAYSCALE)
height, width = inimage.shape

esitlenmis_image = np.zeros((height, width), dtype=np.uint8)
```

```
# Histogramı hesapla
hist = cv2.calcHist([inimage], [0], None, [256], [0, 256])
new_deger = np.zeros((256, 1))

for i in range(256):
    new_deger[i] = sum(hist[:i]) * 255 / (height * width)

for i in range(height):
    for ii in range(width):
        esitlenmis_image[i, ii] = new_deger[inimage[i, ii]]

# Ortalama hesaplama
def ortalama(goruntu):
    toplam = 0
    h, w = goruntu.shape
    for i in range(h):
        for j in range(w):
            toplam += goruntu[i, j]
    return toplam / (h * w)

# Standart sapma hesaplama
def standart_sapma(goruntu, ort):
    toplam = 0
    h, w = goruntu.shape
    for i in range(h):
        for j in range(w):
            fark = goruntu[i, j] - ort
            toplam += fark * fark
    return (toplam / (h * w)) ** 0.5

# Giriş ve çıkış için ortalama ve standart sapma hesapla
giris_ort = ortalama(inimage)
giris_std = standart_sapma(inimage, giris_ort)

cikis_ort = ortalama(esitlenmis_image)
cikis_std = standart_sapma(esitlenmis_image, cikis_ort)

# Sonuçları yazdır
print("Giriş Görüntüsü Ortalama:", giris_ort)
print("Giriş Görüntüsü Standart Sapma:", giris_std)
print("Çıkış Görüntüsü Ortalama:", cikis_ort)
print("Çıkış Görüntüsü Standart Sapma:", cikis_std)

# Görüntüleri göster
plt.subplot(1, 2, 1), plt.imshow(inimage, cmap='gray'), plt.title('Orijinal Görüntü')
plt.subplot(1, 2, 2), plt.imshow(esitlenmis_image, cmap='gray'), plt.title('Eşitlenmiş
Görüntü')
plt.show()

hist1 = cv2.calcHist([esitlenmis_image], [0], None, [256], [0, 256])

plt.plot(hist)
plt.title('Giriş Görüntü Histogramı')
plt.xlabel('Piksel Değeri')
plt.ylabel('Piksel Sayısı')
plt.show()

plt.plot(hist1)
plt.title('Çıkış Görüntü Histogramı')
plt.xlabel('Piksel Değeri')
plt.ylabel('Piksel Sayısı')
plt.show()
```

3.5 Contrast stretching

```
#kontrast germe

import cv2
import numpy as np
import matplotlib.pyplot as plt

# Gri seviye bir örnek görüntü yükleme
inimage = cv2.imread('monalisa.jpg', cv2.IMREAD_GRAYSCALE)

def KontrastGerme(image):

    row, col = inimage.shape
    # Predefine the output image
    normalized_image = np.zeros((row, col), dtype=np.uint8)

    for y in range(row):
        for x in range(col):
            normalized_image[y, x] = (inimage[y,x] -
np.min(inimage))*(255/(np.max(inimage)- np.min(inimage)))

    return normalized_image

Outimage = KontrastGerme(inimage)

# Görüntüleri gösterme
plt.subplot(1, 2, 1), plt.imshow(inimage, cmap='gray'), plt.title('Original Image')
plt.subplot(1, 2, 2), plt.imshow(Outimage, cmap='gray'), plt.title('Kontrast Gerilmiş
Image')
plt.show()
```